(1)

GIT-SERC-86/06

Software Test and Evaluation Manual, Volume 3.

GOOD EXAMPLES of SOFTWARE TESTING

in the

DEPARTMENT OF DEFENSE

Software Test and Evaluation Project

1 October 1986

Prepared for

OUSDRE (T&E)
The Pentagon, Room 3E1060
Washington, D. C. 20301

U. S. ARMY MISSILE COMMAND
ATTN: AMSMI-YCX
Redstone Arsenal, AL 35898

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: GIT-SERC-86/06	2. GOVT ACCESSION NO.: AD-A189 425
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): Good Examples of Software Testing In The Department of Defense	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report
	6. PERFORMING ORG. REPORT NUMBER: GIT-SERC-86/06
7. AUTHOR(s): Software Test & Evaluation Project	8. CONTRACT OR GRANT NUMBER(s): BOA DAAH 01-85-D-A005 D.O. 0012
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Software Test & Evaluation Project, Software Engineering Research Center, GIT, Atlanta, Georgia 30332-0280	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Missile Command, Redstone Arsenal, Alabama 35898	12. REPORT DATE: 10/1/86
	13. NUMBER OF PAGES: 50 + v
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Test and Evaluation; Software Test & Evaluation Project

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report contains the results of a study conducted by the Software Test and Evaluation Project (STEP) to locate and document good examples of software testing of major systems developed by the Department of Defense.

Three systems were chosen for this study: one from the Navy, one from the Air Force, and one from the Army. The systems selected were recommended by organizations within their respective Services. (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data En

UNCLASSIFIED

VI: GIT-ICS-85/26, VII: GIT-SERC-87/03
VIII: GIT-SERC-86/06

SOFTWARE TEST AND EVALUATION MANUAL
VI: GUIDELINES FOR TREATMENT OF SOFTWARE IN TEMPS
VII: GUIDELINES FOR SOFTWARE TEST AND EVALUATION IN THE DOD
VIII: GOOD EXAMPLES OF SOFTWARE TESTING IN THE DOD

FINAL REPORT

SOFTWARE TEST AND EVALUATION PROJECT

F33657-82-G-2083 and
BOA DAA H01-85-D-A005
D.O. 0008 and 0012

SOFTWARE TEST AND EVALUATION PROJECT
SOFTWARE ENGINEERING RESEARCH CENTER
GEORGIA INSTITUTE OF TECHNOLOGY, ATLANTA GA 30332

ADDRE(T&E) RM. 3D1075 THE PENTAGON
DR. H. STEVEN KIMMEL WASHINGTON DC 20301

I:10/85,II:2/87,III:10/86

I:53+iv,II:122+viii,III:50+vi

UNCLASSIFIED

DISTRIBUTION UNLIMITED

DISTRIBUTION UNLIMITED

Software Test and Evaluation Manual, defense system acquisition, Software Test and Evaluation Project (STEP), Test and Evaluation Master Plans (TEMPs), mission critical computer resources, software intensive, risk, checklist, program offices, independent test organizations, operational and technical characteristics, software test tools and resources.

The Software Test and Evaluation Manual is a three volume reference set that provides examples, checklists, and guidance for Department of Defense Components in the area of software test and evaluation for major defense system acquisition. The primary focus of the Manual is to improve the test and evaluation of major systems through improved acquisition management and risk reduction procedures.

The concepts developed during the STEP Project have been incorporated into DoD 5000.3-M-3, "Software Test and Evaluation Manual." This reference set expands on

(continued - over)

UNCLASSIFIED

these concepts and provides additional insight into how software test and evaluation principles can be addressed within the overall system acquisition process.

Volume I, Guidelines for the Treatment of Software in Test and Evaluation Master Plans, was intended to support the review of Test and Evaluation Master Plans (TEMPs) for systems that contain mission critical software components, are software intensive, or present software testing issues that significantly affect risk. It consists of a checklist or series of questions that were keyed to the then existing major paragraphs of a TEMP, and an accompanying set of explanatory notes that are brief commentaries on the questions and the significance of the possible responses to them. Although the keying to the TEMP paragraphs is no longer valid, the checklist does identify software related TEMP issues that are of continuing interest within DoD.

Volume II, Guidelines for Software Test and Evaluation in the Department of Defense, addresses the structuring, planning, conduct, and evaluation of software tests throughout the acquisition process. It is intended for use by the Service Headquarters, Development Commands, Program Offices, Development Test Agencies, Operational Test Agencies, and contractors. It may also be of interest to Software Support Agencies and individuals within User Commands that become involved in requirements definition and evaluation prior to a new system's fielding.

Volume III, Good Examples of Software Testing in the Department of Defense, contains the results of a study to locate and document good examples of software testing in major weapon systems developed by the Services. Three systems were chosen for this study: one from the Army, one from the Navy, and one from the Air Force. When examples of good software testing were located, documentation was sought to substantiate the claim. If a good example was found that was not documented, it was in the report as a subjective observation. This Volume serves as a source of "lessons learned" on software testing for other system development efforts.

TABLE of CONTENTS

SECTION PAGE

SECTION PAGE

SECTION PAGE

1.0 INTRODUCTION

This report contains the results of a study conducted by the Software Test and Evaluation Project (STEP) to locate and document good examples of software testing of major systems developed by the Department of Defense.

Three systems were chosen for this study: one from the Navy, one from the Air Force, and one from the Army. The systems selected were recommended by organizations within their respective Services.

Data on each system was obtained by conducting interviews with the Service Program Office personnel and the Prime Contractor's Program Office personnel (including, if they existed, software managers and software test managers), and reviewing available program documentation.

The interviews were guided by standard questions that were developed to determine whether or not the systems had conducted software testing in a manner that could be used to illustrate how other projects could improve their own software test programs. The questions were based on the recommendations of the Software Test and Evaluation Project (Reference, Software Test and Evaluation Project, Phases I and II Final Report, Volume I, Report and Recommendations, R. A. DeMillo and R. J. Martin). These recommendations for improving software testing had been formulated following the analysis of the results of surveys of the state of the practice and state of the art of software testing in the Department of Defense. In addition, the questions served to promote consistency in data gathering and to initiate dialogue with the personnel interviewed.

When examples of good software testing were located, documentation was sought to substantiate the claim. If an instance of a good example was found that was not documented, it was not discarded but was included in this report as a subjective observation. Keywords: Validation testing, Software integration

This report has been organized into five major sections. Section 2.0, Summary of Findings, includes brief descriptions of the good examples located on each of the three systems. Sections 3.0, 4.0 and 5.0 detail the findings on each system, TACTAS, Pave Paws and Firefinder, respectively. Sections 3.0, 4.0 and 5.0 have been organized into subsections that allow the reader to extract any level of information he feels will be of use to him in his endeavors. The initial information associated with the description of the program and its contractors is contained in the first subsection. The second subsection contains an overview of the program, its software and the testing conducted on the software. The last subsection contains a detailed description of the good examples and subjective observations. In addition, a set of Appendices is included in the report further detailing some of the good examples.

2.0 SUMMARY OF FINDINGS

Each program reviewed did in fact reveal good examples of software testing. The following sections summarize the findings on each program.

2.1 TACTAS

1. The Program Office recognized the need for a separate software manager and established that position prior to contract award.

2. The Program Office contractually required the full application of MIL-STD-1679 (Military Standard, Weapon System Software Development). This Standard includes comprehensive requirements for software testing as well as for software development.

3. The Prime Contractor designated a separate software manager and established an independent software test organization.

4. The Prime Contractor utilized a systematic software integration methodology.

5. The Prime Contractor employed a formal problem reporting and resolution system to track software errors.

6. The Prime Contractor established turnover criteria for delivery of software to the independent test group and to the system's test group.

7. Two other items brought out by the Program Office and the Prime Contractor that they felt contributed to the overall success of the program and therefore influenced the success of the software testing, even though indirectly, were:

   The Program Office felt that the working relationship with the Prime Contractor was excellent.

   The Prime Contractor felt that the Program Office was aware of the increased costs of applying a stringent software development and test methodology and funded the program accordingly.

2.2 PAVE PAWS

1. The Program Office recognized the need for a separate software manager and established that position prior to contract award.

2. The Prime Contractor recognized the need for a separate software manager and established that position at Critical Design Review (CDR). (The Prime Contractor, in fact, planned to establish a software manager from the beginning of the program, but due to personnel problems was not able to establish the position until CDR.)

3. The Prime Contractor initiated early software test planning and the conduct of early testing of critical functions. This is exemplified by the planning and establishment of an in-plant string test facility. The use of the string test facility allowed for early visibility into the functional operation of the software with the deliverable system hardware prior to installation at the site.

4. The major software sub-contractor established an independent test team within its organization.

5. A documented systematic formal approach was used for verifying the software requirements.

6. Three other items brought out by the Program Office and the Prime Contractor that they felt contributed to the overall success of the program and therefore influenced the success of the software testing, even though indirectly, were:

   The major software sub-contractor applied personnel that had just completed development of a similar system.

   The Program Office insisted, but did not contractually require, that all software was to be developed and tested at one location.

   The Program Office ensured prior to contract award that the requirements for the system were complete and met the needs of the user. They felt that this was the prime contributor to the system exhibiting stable requirements throughout its development.

2.3 FIREFINDER

1. The Program Office staff was organized in anticipation of issues arising from a software intensive radar development program. The organization was composed of personnel who had experience in the development of similar radar systems and were aware of the risk of developing software intensive systems.

2. The Program Office commissioned the development of a Radar Environment Simulator (RES) to support system test and integration. The primary use of the RES was to demonstrate a baseline system capability prior to testing it against actual incoming projectiles. The testing scenarios which could be presented by the RES proved useful in the detection of software faults.

3. The Prime Contractor designated an independent Software Test Manager. The responsibilities of the Software Test Manager were to specify programming practices, testing methods, and the degree of testing employed throughout the development effort.

4. The Prime Contractor planned for and utilized instrumentation and simulators specifically intended for software testing activities. This need stemmed from primarily two requirements: the real-time nature of the software system, and the uncertainties in the quality of received radar returns. The first, real-time operation, prompted the development of monitoring instrumentation in the form of a logging tape drive and a dynamic execution monitor display. The second, quality of received signal, prompted the development of a simulator that generated radar inputs for the software.

3.0 TACTAS

3.1 System Data

Name of System:	TACTAS, AN/SQR - 19
Service:	US Navy
Development Organization:	US Navy Naval Sea Systems Command Washington, D. C.
Prime Contractor:	General Electric Co. Underseas Electronics Division Syracuse, NY
Software Sub-Contractors:	None
IV&V Contractor:	None
Documents Reviewed:	AN/SQR-19 Tactical Towed Array Sonar (TACTAS) System Life Cycle Management Plan, 2 November 1983. Software Development Plan for AN/SQR-19 25 October 1979. MIL-STD-1679, Weapon System Software Development (Navy), 1 December 1978

3.2 Development History

3.2.1 Program Description

TACTAS is a towed array passive sonar system that is a part of the US Navy's Antisubmarine Warfare Suite. The system is composed of three subsystems, the Towed Array, the Handling and Stowage Equipment and the Ship-based Electronics. The system is designed for installation on the DD-963 destroyer, the FFG-7 Guided Missile Frigate, and the DDG-47 destroyer. The original program was for the development of three pre-production systems for evaluation. The contract for development was originally let in 1976. The original program was terminated in 1978 and the contract was restarted in 1979. The first system was installed on-ship in the first quarter of FY 1982 with Tech Eval initiated in the fourth quarter of FY 1982. The system is currently in production and is undergoing design modification due to a change in the threat.

3.2.2 Software Description

The software for TACTAS consists of seven Computer Program Configuration Items (CPCIs) which are executed on the computers resident in the Ship-based Electronics Subsystem. The CPCI names, host-target configuration and development language are shown in Table 3.2-1

CPCI	Host Machine	Target Machine	Language
1-AN/UYS-1 Analyzer Unit Computer Program	FASP See Note 1	AN/UYS-1	SPL/1
2-AN/UYK-20 Automatic Detection and Tracking Computer Program Software	AN/UYK 20	AN/UYK 20	Assembler
3-AN/UYK-20 Bearing Track Computer Program	AN/UYK 20	AN/UYK 20	Assembler
4-AN/UYK-20 Display and Control Computer Program	AN/UYK 20	AN/UYK 20	Assembler
5-AN/UYK-20 Real Time Monitor Computer Program	AN/UYK 20	AN/UYK 20	Assembler
6-AN/UYK-20 Initialize and Load Computer Program	AN/UYK 20	AN/UYK 20	Assembler
7-Signal Conditioner and Receiver Diagnostics Computer Program	AN/UYK 20	AN/UYK 20	Assembler

TABLE 3.2-1

Note 1: The Facility for Automated Software Production (FASP) is resident at the Naval Air Development Center, Warminster, PA. FASP was used for the development of the software resident on the AN/UYS-1. FASP was accessed via a remote job entry station at the contractor's facility.

3.2.3 Software Test Overview

Software testing consisted of informal unit/module and build testing, and formal Validation and Software Integration conducted in accordance with MIL-STD-1679. The requirements of MIL-STD-1679 were used as a guide for the unit/module testing. The build testing was conducted using a top down strategy where builds were defined using functional capability lists to direct the integration of units into the test bed.

The Prime Contractor established three facilities for development and test of software, and integration and test of software and hardware.

3.3 Analysis of Explicit Good Examples

This section reviews, in detail, items of potential use as examples of improved planning and conduct of software testing.

3.3.1 The Program Office Designated a Separate Software Manager

The Program Office designated a separate software manager from the inception of the contracted portion of the program. The software manager was responsible for the overall software development on the program. He was assisted in his task by engineers assigned to the program from the Naval Undersea Systems Center.

3.3.2 The Program Office Contractually Required the Full Application of the Navy Software Development Standard (MIL-STD-1679)

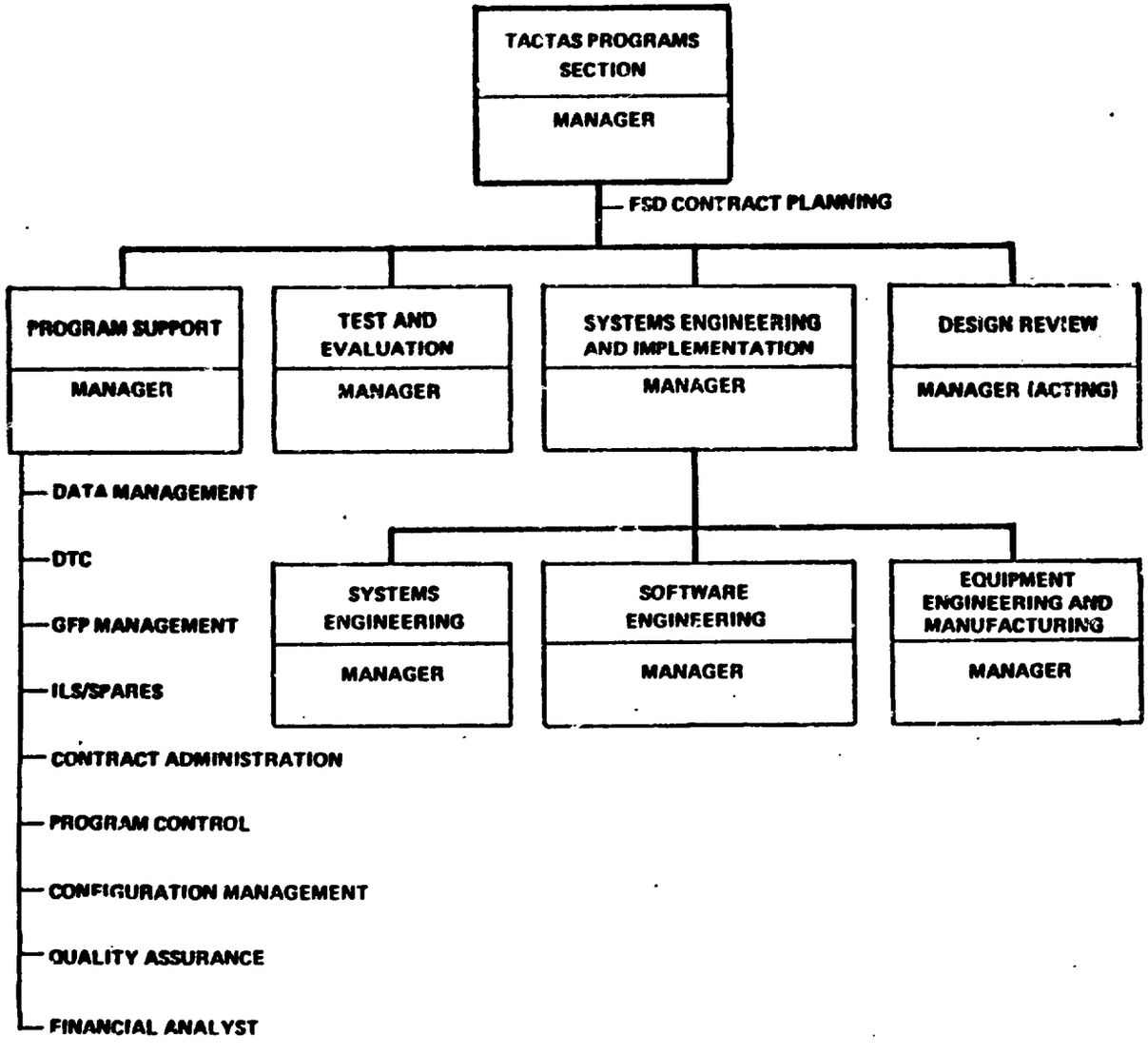
The TACTAS contract required the full application of MIL-STD-1679 to the software development. Included in that standard are requirements for rigorous software testing. The Prime Contractor detailed the implementation of those requirements in its Software Development Plan. Appendix A contains the description of the Prime Contractor's implementation of those requirements relevant to software testing.

3.3.3 The Prime Contractor Designated a Separate Software Manager and an Independent Software Test Organization

The Prime Contractor recognized the need for an Independent Software Test Organization (called Program Test) for a system of the complexity of TACTAS. The organization of TACTAS and the relationship of Software Engineering to the program is shown in Figure 3.3-1. The organization of Software Engineering including Program Test is depicted in Figure 3.3-2. The Program Test Organization was responsible for the Computer Program Test Plan and build, integration and validation test specifications, procedures and reports. The Program Test group also identified requirements for software test drivers to be used during build, integration and validation tests, and conducted the build, integration and validation testing. Upon completion of validation and integration testing and closure of all software problem reports, the Program Test group turned the software package over to Test & Evaluation (T&E) Engineering and supported subsequent test activities. The Program Test group also maintained a Program Performance Specification (PPS) requirements traceability matrix for each computer program that traced test cases to PPS requirements to assure that all software functional requirements were verified.

In addition to the direct responsibilities of the Program Test organization the T&E Engineering Organization was responsible for subsystem and system integration testing of this software intensive system. The following paragraphs, extracted from the Software Development Plan describe the organizational relationships between Systems Engineering, Software Engineering and T&E Engineering:

> Figure 3.3-2 depicts the design/development/test responsibilities for the AN/SQR-19 operational computer programs. AN/SQR-19 Systems Engineering develops computer program performance and test requirements and documents these requirements in PPS's. Systems Engineering has developed a data processing/display simulation in the Sonar Evaluation and Demonstration Model (SEDM) facility for evaluation of selected signal and data processing display modes. In addition, Systems Engineering performed testing of operational computer program algorithms to insure that system performance requirements will be satisfied by the software.



79-34

Figure 3.3-1 AN/SQR-19 Program Management Organization

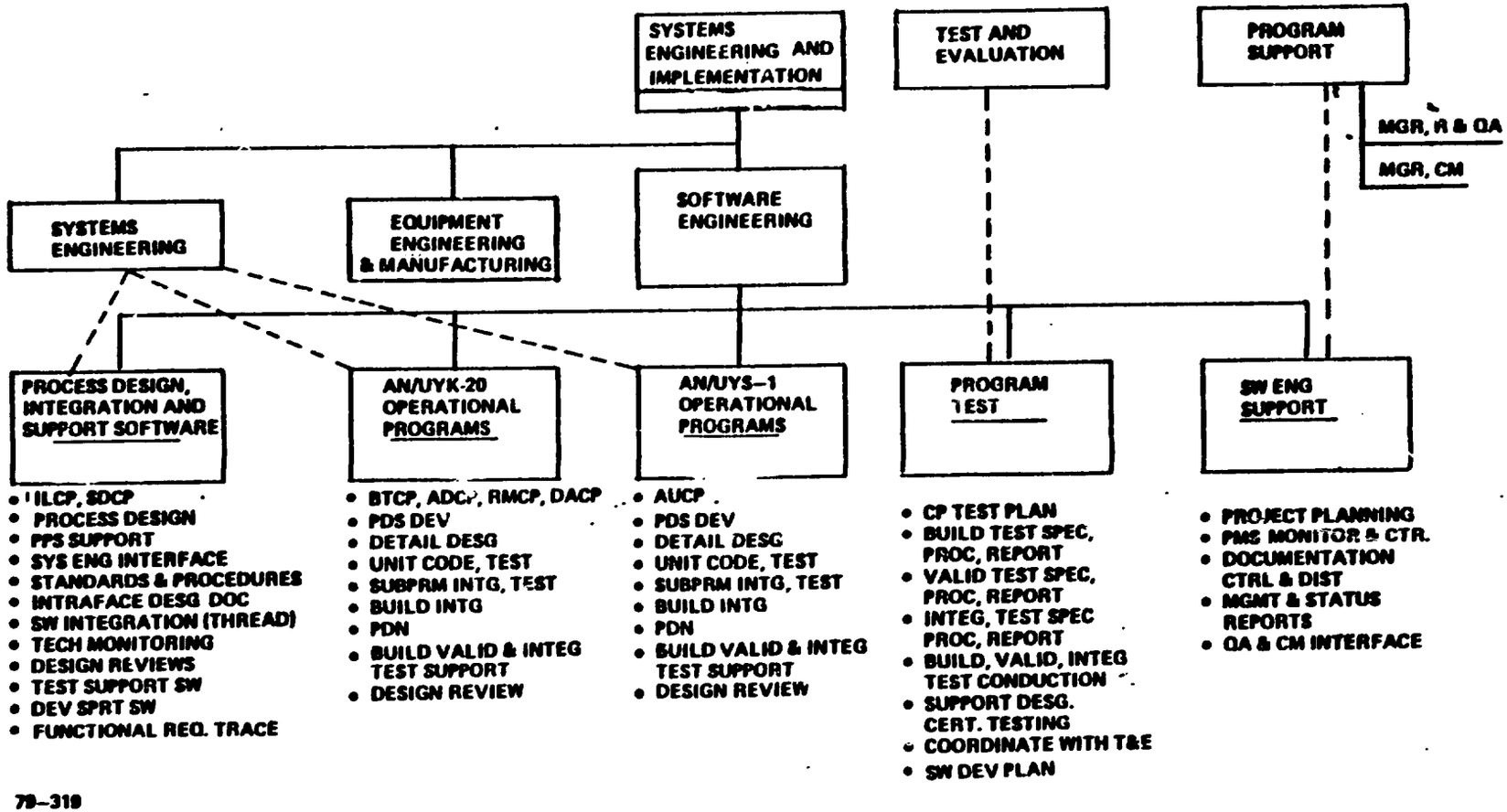


Figure 3.3-2 AN/SQR-19 Software Engineering Organization

Software Engineering has the responsibility for the design, test, development, and integration of operational computer programs to satisfy the requirements specified in the PPS's. This responsibility includes the development of Program Design Specifications (PDS's) which include the Common Data Base Design Document (CDBDD), and the design, code, and test of computer programs. The detailed design of the software is documented in the Program Development Notebooks (PDN's) and detailed interface information is recorded in the Interface Design Document (IDD). Test responsibility includes demonstration of compliance with applicable PPS and PDS requirements. Software Engineering will also support Systems Engineering in generation of PPS requirements.

Software Engineering is responsible for the development of build, validation, and software integration test specifications, test procedures and test reports. Software Engineering will specify and develop tests, including required test software, for build, validation and integration testing.

T&E will be responsible for developing test specifications, procedures and reports at the SES subsystem and system test levels. T&E also has the responsibility for three AN/SQR-19 test facilities. These test facilities consist of a Software Development Facility (SDF) a Ship Based Electronics Test Facility (STF), and an Integrated Test Facility (ITF). The SDF and the STF test facilities will contain a complete TACTAS complement of AN/UYS-1 and AN/UYK-20 computers and peripheral equipment to support SES integration and subsystem design certification testing. Software Engineering will support T&E by providing test software and assisting T&E in the Hardware/Software Integration testing activities in the SDF and ITF facilities.

As depicted in Figure 3.3-2 the Software Quality Assurance organization will perform periodic reviews and audits of software requirements, design code and test results. This activity is reflected in the software QA review of Program Performance Specifications, Program Design Specifications, Program Development Notebooks and test documents. Software QA will also witness selected software tests.

3.3.4 The Prime Contractor Used a Systematic Build Test Methodology

The build test approach used on TACTAS consisted of a top down technique of incremental builds. The methodology was explained in detail in the Software Development Plan and is included as Appendix B for reference. The description is comprehensive, in that it addresses the total software test approach including build testing. The build testing itself concentrated on testing strings of integrated software units and their interfaces. The following paragraphs were extracted from the Software Development Plan and Appendix B to describe the overall build test approach.

> Top down testing using a series of incremental builds is the key element in the approach to development and validation of each of the AN/SQR-19 CPCI's. This approach has been proven to be an efficient method of integrating software units and subprograms into a working CPCI and verifying that performance and design requirements have been met.
>
> The preliminary software build, which is used as a test bed by the Software Design Team for lower level unit and subprogram testing, is called the dummy process. The dummy process initially contains dummy stubs for all application subprograms which are subsequently replaced by the actual code through a series of incremental software builds. Build 1 testing, for example, is conducted for each CPCI by the software test team when lower level tests are completed by the design team on the units and subprograms comprising the build 1 capability. As described in subsequent paragraphs, the build 1 capability will be augmented in subsequent builds by additional code deliveries for each CPCI. At the conclusion of the final build test (which is conducted with all subprograms in a given CPCI), validation testing will be performed on the CPCI.

3.3.5 The Prime Contractor Employed a Formal Problem Reporting and Resolution System to Track Software Errors

The Software Development Plan detailed the use of a formal reporting and resolution system for tracking software errors. Error reporting was initiated when software was turned over for build testing. All types (documentation and requirements, as well as coding errors) were tracked during all phases of testing. Each problem report was processed by the Configuration Control Board and software quality assurance audited the process.

3.3.6 The Prime Contractor Employed Turnover Criteria to Deliver Software to the Software Test Group and then from the Software Test Group to T&E Engineering

There were two levels of turnover tests conducted.

1) The first turnover test was conducted by the developers, to deliver "builds" to the Software Test group for independent testing.

   A build turnover test with clearly specified pass/fail criteria was designed by the Software Test group with participation by the development group responsible for the software. The idea of the turnover test was to establish the sanity of the build in the test bed. This meant that the software cycled and that a few high level capabilities were satisfied in order to facilitate further testing. The pass criteria was not rigorous for turnover tests. The strategy was for the developer to pass the turnover test in the development test bed, then deliver the software to the test team.

   Once the test team certified that the software had passed the turnover test, it became their responsibility and the development teams, from that point on, only needed to respond to software problem reports against that software.

2) The second turnover test was conducted at the completion of integration thread testing and validation testing by the software test team. The purpose of this test was to turnover the integrated Computer Program Configuration Items (CPCI's) to T&E Engineering for conduct of the Subsystem Design Certification Tests.

   At the completion of integration thread testing and validation testing the AN/SQR-19 software was turned over to T&E Engineering for subsystem Design Certification testing.

   The completion criteria for validation testing was based on the review of all validation test documentation including test reports by the Software Test Review Board.

A Software Test Review Board was established to verify that all planned tests had been performed and that the results of these tests reflected successful accomplishment of validation tests as described by the Software Validation Test Specifications, Test Procedures, Computer Program Test Plan, and Section 4.0 of the PPS's, as appropriate. This board consisted of the following members:

a) Manager, AN/SQR-19 Software Engineering - Chairman
b) Project Engineer, Software Test
c) Project Engineer, Process Design Integration & Support Software
d) Systems Engineering representative
e) Test and Evaluation Engineering representative
f) Computer Software Reliability and Quality Assurance representative
g) Navy representative

Membership on the board may have also included ad hoc representatives from the development and test teams. Representation on the board varied as a function of the CPCI test results being reviewed. In order to facilitate the test planning and review process, the software test team generated a test case matrix for each CPCI which traced PPS performance requirements to the code to be tested in each test case.

The completion of integration testing was based on verification of interface performance requirements in the PPS, the Interface Design Document (IDD) and partially the Interface Design Specification (IDS). The test plans and procedures, but not reports, were reviewed by the Software Test Review Board.

3.4 Analysis of Subjective Examples

The following items which were brought out by the Program Office or the Prime Contractor were felt by them to have contributed to the overall success of the program. Since no explicit documentation of these items exists and because of their subjective nature, these items are separated from the previous items. The implementation of some of these suggestions can also be categorized as good management, an unquantifiable parameter. But, it should be emphasized that these items probably contributed a great deal to the success of the program and should be used as goals on similar developments.

3.4.1 The Importance of Customer/Contractor Relations

The Program Office felt that one of the keys to the success of TACTAS was the good working relationship that was established between the Navy engineers and the Prime Contractor's engineers. The Program Office felt that communication was open and frank, thus allowing the separation of real problems from imagined ones.

3.4.2 The Need for Adequate Funding for Software Development and Testing

The Prime Contractor felt that the Navy recognized that developing software as rigorously and formally as required by MIL-STD-1679 was costly, budgeted accordingly and accepted the contractor's costs for using this approach.

4.0 PAVE PAWS

4.1 System Data

Name of System:	PAVE PAWS, AN/FPS-115
Service:	US Air Force
User:	US Air Force Space Command
Development Organization:	US Air Force Electronics Systems Division Hanscom Air Force Base, Mass.
Prime Contractor:	Raytheon Equipment Division Wayland, MA.
Software Sub-Contractors:	IBM Federal Systems Division Gaithersburg, MD Control Data Corporation Minneapolis, MN Raytheon Missile Division Bedford, MA
IV&V Contractor:	None
Documents Reviewed:	System Performance Specification for Phased Array Warning System Radar Set-AN/FPS-115, SS-OCLU-75-1A, 15 December 1977, revised 28 March 1980. Pave Paws Computer Program Development Plan, ER 76-4110, 26 January 1977. Statement of Work for Pave Paws, 4 April 1975, revised 10 May 1975.

4.2 Development History

4.2.1 Program Description

PAVE PAWS is a large phased array radar system used for early warning of Submarine Launched Ballistic Missiles. It has a secondary mission of supporting the Space Track Mission of the US Air Force SPACETRACK System. The Pave Paws system interfaces with the Norad Cheyenne Mountain Complex, the National Military Command Center, the Strategic Air Command and the Alternate National Military Command Center.

The original contract called for the development and installation of two radar systems at Otis AFB, Mass. and Beale AFB, Calif. These two systems were to be developed and installed in 48 months, with the first system scheduled to be installed and operational in 36 months. The development contract for the two systems was let in May 1976. The Otis site was accepted on schedule by the Air Force on April, 1979; the Beale site was delivered ahead of schedule on February, 1980.

Currently, two additional sites are under development in Georgia and Texas to complete the planned four site network; these will become operational in November, 1986 and May, 1987, respectively. A third phase of the program to upgrade the whole network, is planned for implementation from 1987 to 1990.

4.2.2 Software Description

The software consists of seven Computer Program Configuration Items (CPCIs). The CPCI names, development contractor, host-target configuration and development language are shown in Table 4.2-1.

CPCI	Development Contractor	Host Machine	Target Machine	Language
1-Cyber Support Software Modifications	CDC	Cyber 174	Cyber 174	Assem.
2-Tactical Applications Software	IBM	Cyber 174	Cyber 174	JOVIAL & Assem.
3-Simulation Software	IBM	Cyber 174	Cyber 174	JOVIAL
4-Structured Programming Dev. Tools	IBM	Cyber 174	Cyber 174	JOVIAL
5-Data Reduction Tools	IBM	Cyber 174	Cyber 174	JOVIAL
6-Radar Control Software	Raytheon Equipment Division	Modcomp IV/25	Modcomp IV/25	FORTRAN & Assem.
7-Signal Processor Software	Raytheon Missile Systems Div.	Cyber 174	Special Purpose Hardware	Assembler

TABLE 4.2-1

The approximate size of the developed software for Pave Paws is 198K source lines of code. Four development organizations were used to produce the software. Control Data Corporation, the main frame data processing equipment vendor, developed the modifications to its Cyber Operating System. IBM was responsible for the mainframe resident CPCIs, other than the Cyber Operating System. Raytheon was responsible for the Radar Control CPCI resident on a minicomputer and the Signal Processing CPCI resident on a special purpose signal processor. Raytheon was also responsible for the development of the B5's (Software Requirement Specifications) and was assisted in that effort by IBM. Of the seven CPCI's developed for the system, three of them (Simulation, Data Reduction and Structured Programming Development Tools) operate off-line on the backup computer system.

4.2.3 Software Test Overview

Software testing consisted of informal unit/module testing and formal Preliminary and Final Qualification Tests as defined in Air Force Regulation 800-14, Acquisition Management, Management of Computer Resources in Systems, 12 September 1975, (see paragraph 4.3.5 for the System Specification requirements to implement the regulations of AFR 800-14). Specific unit/module testing was not contractually required and was left to the contractor's discretion. The application of a specific software test strategy and/or technique was not found at the unit/module level. At the CPCI level, a top-down testing strategy was required by the Computer Program Development Plan (CPDP). Simulators were used to test integrated software prior to its integration with the radar system as stated in the CPDP.

The Prime Contractor proposed and implemented the use of a string test facility in his plant for hardware/software integration. The string test facility consisted of all system equipment with the exception of the full phased array and associated transmitters (a subarray was used).

4.3 Analysis of Explicit Good Examples

This section reviews, in detail, items of potential use as examples of improved planning and conduct of software testing.

4.3.1 The Program Office Established a Separate Software Manager

The Program Office designated a separate software manager from the inception of the contracted portion of the program. The software manager (an Air Force Major) was responsible for the review and approval of all aspects of the software development for the Air Force, including the Software Requirements Specifications (B5's) and the Formal Qualification Testing.

4.3.2 The Prime Contractor Designated a Software Manager

The Prime Contractor designated the Deputy Program Manager as responsible for the complete software development. The organization of the Pave Paws Software Development is shown in Figure 4.3-1.

4.3.3 The Prime Contractor Established a String Test Facility

The Prime Contractor proposed and established a string test facility, in-plant, to be used for integration and test of the system software and hardware prior to installation at the site. (A string test facility is defined, in this context, as a facility consisting of the operational hardware and software which is used to integrate and test those items prior to site installation.) The Program Office and the Prime Contractor felt that this helped avoid many of the problems associated with testing a system of this size and complexity. Namely, the complete development team was available to assist in problem solving, the string test allowed for early visibility into hardware/software integration and the complete software development facility was available to assist the developers in resolving and correcting errors.

In particular, the proposal and use of a string test facility is a good example of early test planning. This indicated the Program Office's and contractor's concern with software integration and test in the total system environment.

The contract (Statement of Work or System Specification) did not explicitly require the use of a string test facility. The Systems Specification did require, "Subsystem tests/integration to the largest scale practicable shall be conducted at the contractor's plant prior to shipment.", (ref. System Specification, paragraph 4.1).

4.3.4 The Major Software Subcontractor Established an Independent Test Team

The subcontractor designated a support software and test manager in its organization. The test responsibilities of the manager and his group included (ref. Pave Paws Computer Program Development Plan):

a. Preparing and presenting the test plans for the Pave Paws software at System Design Review (SDR), Preliminary Design Review (PDR), and Critical Design Review (CDR).
b. Defining all test cases, schedules, procedures, expected results, documentation and classes of tests.

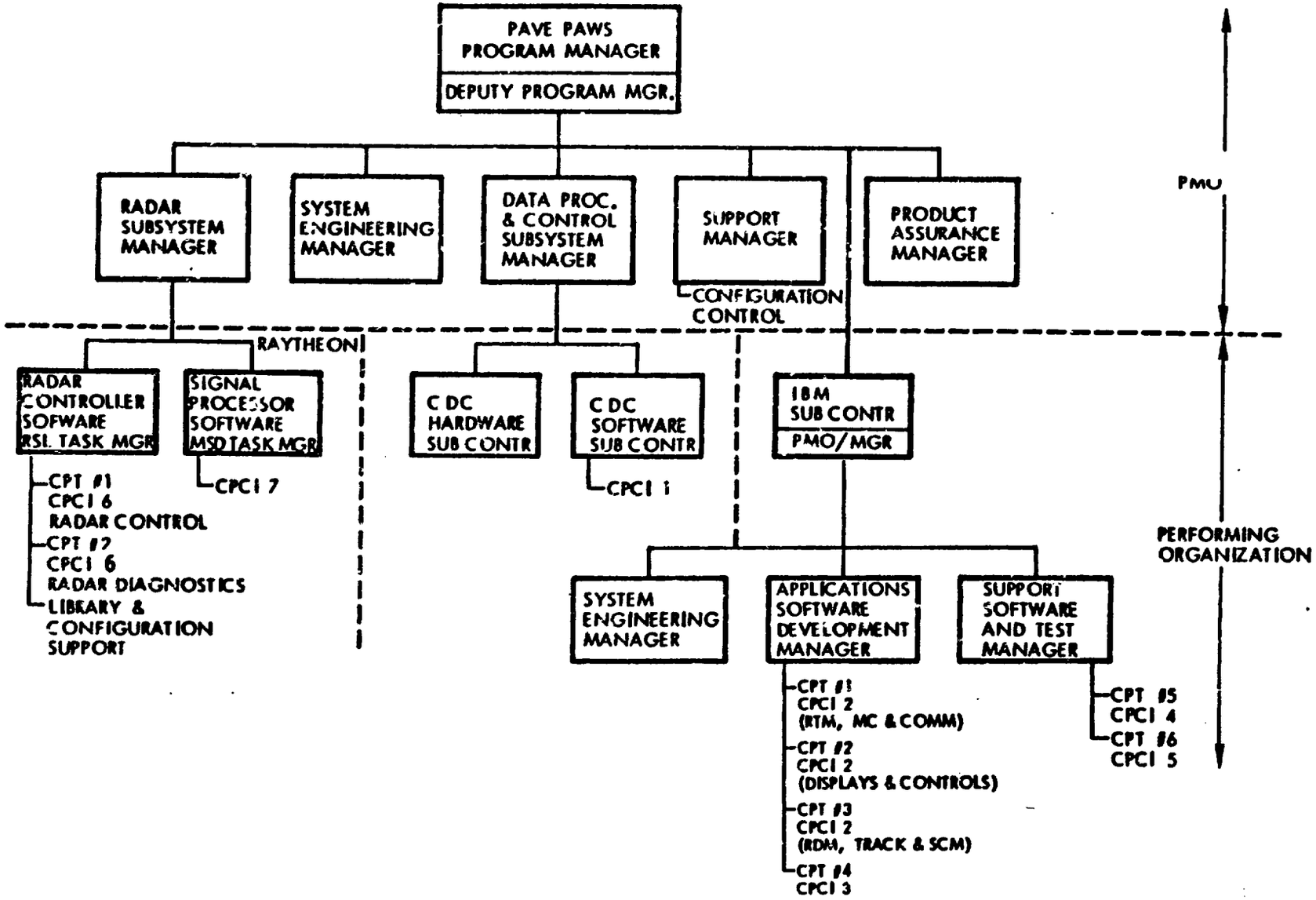


Figure 4.3-1 PAVE PAWS Program Organization For Software

c. Defining all test data requirements so appropriate interface stub programs and environmental simulators were available at test execution time.
d. Defining and generating all scenarios required for testing, utilizing the off-line scenario generator when available.
e. Writing all test specifications, procedures and reports for all qualification tests under his direct control.
f. Providing support for integration testing at both the subsystem and system levels.

4.3.5 A Documented Systematic Formal Approach was Used for Testing the Software at the Requirements Level

The System Specification for Pave Paws (System Performance Specification for Phased Array Warning System, Radar Set AN/FPS-115, SS-OCLU-75-1A, 28 March 1980) contained the requirements of this testing. The regulations of AFR 800-14 for Qualification Tests were implemented through these requirements. The System Specification requirements are repeated below:

a. Software Qualification Tests. Preliminary and formal qualification tests shall be conducted to verify compliance with contracted specifications and to validate that the software performance and design is in accordance with the requirements specified in Section 3. Preliminary qualification tests shall be conducted in the contractor's plant. Formal qualification tests may be conducted in-plant or on site.

b. Preliminary Qualification Tests (PQT). The Computer Program Components (CPC) which together constitute each Computer Program CI (CPCI) shall be thoroughly tested to establish that each operates correctly, in conformance with the applicable functional requirements specified in Section 3 of this specification, and those in the relevant CPCI specifications. These tests shall be conducted incrementally as each component (or group of components) is developed and checked out. The test sequence shall reflect the development sequence. As components are tested, they shall be used to support the testing of the other functions. The full PQT shall be structured to accord with an approved test plan. The test shall be conducted in accordance with test procedures which define each test, including all inputs and predicted outputs.

b. Formal Qualification Tests (FQT). Formal qualification tests of software shall be conducted after satisfactory completion of preliminary qualification tests. Formal qualification tests shall be conducted in as near a operational environment as possible using both actual and simulated input data. The purposes of formal qualification tests shall be to verify that the software CIs which were subjected to preliminary qualification tests satisfy their specified requirements, that the software CIs can be installed in the operational system, and that each software CI is functionally compatible and satisfies the requirements specified herein. All specified requirements shall be verified as part of FQT.

4.3.6 The Use of Testing Tools on Pave Paws

The use of special test tools on Pave Paws was restricted to the Program Support Library (the PSL had a capability to generate stubs to simulate calls and returns of missing modules during integration testing) and Data Reduction Programs. These items (the PSL and Data Reduction Programs) were called out as requirements of the System Specification (see the following paragraphs for the System Specification Requirements of these items) and therefore, funded as a part of the contract.

a. Structured Programming Development Tools. Structured Programming Development Tools shall be used to provide necessary tools for development and management of the PAVE PAWS Tactical Applications Software, Simulation software, Data Reduction Tools, and shall be used where practical for the Radar Control Software.

The software development shall make use of the following software development tools.

1. Program Support Library

2. JOVIAL language Pre-compiler

3. COMPASS Language Pre-compiler (COMPASS is the assembly language of the CDC Cyber 174)

4. Management Report Generation

b. Data Reduction. Data Reduction Tools provide the off-line Data Reduction software required to read recording tapes written by the Tactical Application and Simulation software and also provide printed output for all recorded data. This shall be in the form of a "block format" which contains descriptive mnemonics for the data in each block printed. A select number of the high usage recording streams will require specialized report programs to provide tabular reports of summaries. Capabilities shall include the following:

1. Time Slicing
2. LRID Selection
   - Do Not Process
   - Format & Print
   - Perform Special Processing
   - Conditional Data Processing
3. Data Conversion/Interpretation
4. Error Processing
5. Call User Supplied Programs
6. List Control Cards
7. Write to Tape or Disk
8. 15 Special Reports

In addition, the Simulation Software CPCI was designed to be used for radar subsystem integration and system integration tests as well as for support of operational exercises. The System Specification paragraph requirements for the Simulation Function is as follows:

a. Simulation. Simulation shall be a detailed system simulation capable of incorporating all of the combinations of missions and threat conditions and radar performance specified herein. The simulation program shall operate on line concurrently with the operational, monitoring, and calibration programs. The entire package shall be capable of operating solely within the data processing facility furnished with the PAVE PAWS and require no other system hardware for execution.

Specific simulation areas shall include as a minimum:

1. Surveillance generation.
2. Known and unknown object handling.
3. Mission control.
4. Mission scenarios.
5. Command and control functions, remote/local.
6. Target capacity and entry rates.
7. Digital communication interface(s) and format(s).
8. Tracking and data collection.
9. Critical overloads.
10. Outputs to training display console.

4.4 Analysis of Subjective Examples

The following items which were brought out by the Program Office or the Prime Contractor were felt by them to have contributed to the overall success of the program. Since no explicit documentation of these items exists and because of their subjective nature, these items are separated from the previous items. The implementation of some of these suggestions can also be categorized as good management, an unquantifiable parameter. But, it should be emphasized that these items probably contributed a great deal to the success of the program and should be used as goals on similar developments.

4.4.1 System Requirements Stability Contributed to the Overall Success of the Program

The Program Office felt that the requirements for the system and software were kept very stable throughout the development. The Program Office felt that the minimization of changes contributed to the overall success in meeting the schedules imposed on the system and the Prime Contractor. The Program Office said they "exhaustively worked" the requirements with the Users prior to issuing the contract to insure the system would meet the User's needs. The Program Office also felt that by defining a simple and small interface to the systems Pave Paws was to interface with, they minimized changes to the system during its development.

4.4.2 The Collocation of the Prime Contractor and the Sub-Contractors Benefited the Test Process

The Program Office felt that one of the key items that contributed to the success of the development of the software for the system was that the software development subcontractors were colocated with the Prime Contractor. They felt in particular that this minimized communications problems, enhanced cooperation and contributed to the overall ease of the execution of the test program. Note that this was not a contractual requirement, but was strongly suggested by the Program Office and endorsed by the Prime Contractor.

4.4.3 The Software Subcontractor Choice Enhanced the Overall Success of the Program

The Program Office felt that the choice of the major software sub-contractor was a good one. The major software sub-contractor (IBM) had recently completed a contract with the Air Force for guidebooks for top-down structured software development and the Program Office felt that it was interested in proving that its recommendations would improve software development. This was therefore believed to have improved the management attention applied to the contract.

4.4.4 The Software Subcontractor's Personnel had Experience in Developing and Testing Similar Software

The Prime Contractor felt that the major subcontractor they selected for the program brought with it a significant level of relevant experience in building and testing systems like Pave Paws. The people assigned to the program by the major software subcontractor had just completed the development and test of similar phased array radar systems for the U. S. Army, the Safeguard Anti-Ballistic Missile Defense System. The timing of the Pave Paws contract allowed the direct transfer of the Safeguard team to the Pave Paws program.

5.0 FIREFINDER

5.1 System Data

Name of System:	FIREFINDER, AN/TPQ - 36
Service:	US Army
Development Organization:	US Army Electronics Research and Development Command Ft. Monmouth, NJ 07703
Prime Contractor:	Hughes Aircraft Company Ground Systems Group Fullerton, CA 92634
Software Sub-Contractors:	None
IV&V Contractor:	None
Documents Reviewed:	Firefinder Computer Program Configuration Management Procedure 22 May 1985 Computer Program Users's Manual for RADAR SET AN/TPQ-37(V) March 1981 Software Production Process 1 September 1979 Programming Standards and Conventions 12 January 1978

5.2 Development History

5.2.1 Program Description

The mission of Firefinder is to detect and track artillery shells and rockets of differing sizes and trajectories. The detection is performed in the presence of high ground clutter and many other targets. When detections are made, tracks are established and the computer backtracks along the trajectories to locate the launch point.

Development of the Firefinder Radars was initiated in 1971. Previous experience associated with the development of the TPQ-28, a predecessor system, had alerted the Army to the difficulties associated with the design and test of a software intensive system and the technical complexities of this type of radar target discrimination.

In 1978, a production contract was awarded. Currently the system is being fielded and the software maintenance activity is being transferred from the prime contractor to the U.S. Army at Ft. Sill.

5.2.2 Software Description

The software for Firefinder consists of three major program groups: Operational Programs (OP), Diagnostic Programs (DP), and Support Programs (SP). The OP provides the control and data processing to detect artillery projectiles and determine the weapon location. The DP are run while the OP is inactive to evaluate radar operation and assist in radar maintenance. The SP is used to support development and modification of OP and DP computer programs. The CPCI names, host-target configuration and development language are shown in Table 5.2-1.

The AN/TPQ-36 software represents approximately 400K lines of ULTRA-16 code. ULTRA-16 is the assembly language for the AN/UYK-15 computer (manufactured by The Sperry Corporation). The present target processor is a Hughes computer product which emulates the AN/UYK-15.

CPCI	Host Machine	Target Machine	Language
1-Operational Programs	AN/UYK-15	AN/UYK-15	ULTRA-16 (assm.)
2-Diagnostic Programs	AN/UYK-15	AN/UYK-15	ULTRA-16 (assm.)
3-Support Programs	AN/UYK-15	AN/UYK-15	ULTRA-16 (assm.)

TABLE 5.2-1

5.2.3 Software Test Overview

Software test requirements for Firefinder were not contractually specified. Software testing was informally conducted at two levels: unit testing and integration testing. The unit level tests were conducted by the development programmers and integration testing was performed under the direction of an independent Software Test Manager.

The application of specific test techniques varied according to the unit's functionality. The test cases were constructed from a software perspective using knowledge of the typical errors associated with either the structures implemented or the implementation language. The unit level test results were evaluated for consistency with the software specification. Some items of the specification were more rigorously evaluated due to the perceived risks associated with the item.

The Software Test Manager specified a turn-over criteria for unit testing prior to that units incorporation into the integration testing configuration. The turn-over, and integration testing criteria were based on the software requirements.

Test cases were generated with the aid of several simulators. A mainframe hosted simulator was used to produce tapes of test data for unit and integration testing. A Radar Environment Simulator (RES) was used to generate test scenarios (target and clutter) for system testing prior to live fire testing. The in-plant system test using the RES was a contractual requirement. The RES was also used as a signal source for integration and system testing.

Test reports for integration activities were produced and problem reports generated. The software related problems were conveyed to the cognizant programmer either verbally or in written form. As software changes were made the units were re-tested by executing the integration test that had uncovered the problem.

5.3 Analysis of Findings

This section reviews, in detail, items of potential use as examples to improve planning and conduct of software testing. The evidence of these examples was derived from verbal discussions and the review of documents generated subsequent to the development activities. It should be noted that formal documentation of these examples has not been discovered, and in all likelihood does not exist, and therefore documented excerpts of their actual implementation has not been included.

5.3.1 The Program Office Staff was Organized in Anticipation of Issues Arising From a Software Intensive Radar Development Program

Analysis of the system requirements for Firefinder, as well as the experience of developing the TPQ-28, had alerted the Army to the fact that a great deal of the functionality of the system would be allocated to the software. This prompted the Program Office to structure its organization to minimize the risk of software's contribution to the success of the development.

The Army established a Technical Management Group in the Program Office that was composed of individuals from within the Army Electronics Research and Development Command. These individuals were experienced in the development of radars of this type and, in addition, they had an appreciation of the risks of implementing a software intensive system.

The Program Office also assigned a person to reside at the contractors facility as a local technical point of contact.

5.3.2 The Program Office Commissioned the Development of a Radar Environment Simulator (RES) to Support System Test and Integration

The primary use of the RES was to demonstrate a baseline system capability prior to testing it against actual incoming projectiles. The initial motivation for such a capability was to reduce the probability of conducting a needless operational test. This concern resulted from a previous development effort for a system which had similar operational requirements as Firefinder. During that development effort the most revealing test cases were presented in an operational environment. The system deficiencies recognized in that phase of testing could have been better remedied if found earlier in development.

The RES was available at the contractors facility for this initial step of the acceptance test process. Because of the capabilities furnished by the RES and its availability to the contractor, it was used by the contractor for development testing as well. The testing scenarios which could be presented by the RES proved useful in the detection of software faults.

5.3.3 The Prime Contractor Designated an Independent Software Test Manager

The Prime Contractor recognized the need for an independent Software Test Manager. The responsibilities of the Software Test Manager were to specify programming practices, testing methods, and the degree of testing employed throughout the development effort. The specified programming practices had been used previously by the prime contractor for software development.

5.3.4 The Prime Contractor Planned For and Utilized Instrumentation and Simulators Specifically Intended for Software Testing Activities

The need for specialized test tools was recognized early in the development program. This need stemmed from primarily two requirements: the real-time nature of the software system, and the uncertainties in the quality of received radar returns. The first, real-time operation, prompted the development of monitoring instrumentation in the form of a logging tape drive and a dynamic execution monitor display. The second, quality of received signal, prompted the development of a software simulated radar signal source.

5.3.4.1 Software Test Instrumentation

A data recording capability was developed which allowed the gathering of execution information on-line for post-test analysis. The system contained a magnetic tape cartridge device which was used as the recording medium. Selected areas of memory could be recorded at either critical points of execution or at periodic intervals. After data gathering was complete the recorded information could be printed in a format to facilitate analysis. Turn around time from test run to data availability was typically one-half day.

A dynamic execution display monitor was developed to allow an immediate assessment of system operation. A portable display unit was attached to the processing electronics to facilitate data presentation. The display could be updated periodically or at critical execution points to display selected areas of memory. The resulting display provided an immediate indication of the changing system state during unit and integration testing.

5.3.4.2 Software Test Simulator

A radar simulator, in addition to the RES, was developed for use in support of software testing. The simulator ran on a mainframe and provided a source of simulated radar return data. The basic function of the simulator was to provide a series of data values representing the trajectory of a projectile. The resulting data values would be recorded on a magnetic tape cartridge for transport to the target processor. The simulator output consisted of values representing track data that were provided to the tracking software of the system. The test data could be modified to represent variations in received signal that were expected in the operational environment. These variations simulated quantization errors and unstable or noisy radar receivers. There was also a provision for varying the presented data in real-time based on the actions of the unit under test. This capability proved useful in testing the beam steer and tracking group integration. In this case the beam positioning outputs caused a data modification suitable to simulate an off-boresight signal return.

Appendix A

The Implementation of the Test Requirements of MIL-STD-1679 on TACTAS

1.0 Introduction

The section is organized to relate the MIL-STD-1679 requirements for software testing to the TACTAS Prime Contractor's Software Development Plan (SDP) responses. Each MIL-STD-1679 requirement is stated followed by the section or sections of the SDP that implemented the requirement.

2.0 Definitions

The SDP, in some cases, uses terminology not defined or used in MIL-STD-1679. The terms and their relation to MIL-STD-1679 terms follow:

2.1 MIL-STD-1679 definition of software components.

A program refers to the weapon system software or one independent part of the software of a weapon system.

Subprogram refers to a major functional subset of a program and is made up of one or more modules.

A module is an independently compilable software component.

Modules are composed of one or more procedures or routines.

2.2 Correlation of MIL-STD-1679 definitions and SDP terms.

The SDP uses the term unit to bracket the MIL-STD-1679 terms of modules, procedures and routines.

The SDP in Section 4.2, Unit Testing (described below) describes both Unit (as defined above) and Subprogram Testing. The separate requirements of MIL-STD-1679 (Module and Subprogram Testing) were met by this section.

The following table relates the MIL-STD-1679 test levels to the test levels described in the SDP.

SDP Test Level	MIL-STD-1679 Test Level
Unit and Subprogram Testing	Module Tests
Unit and Subprogram Testing	Subprogram Tests
Build Testing	Program Performance Tests
Validation Testing	Program Performance Tests
Integration Testing	System Integration Tests
Subsystem Design Certification Testing	No MIL-STD-1679 Test Requirement (See Note 1) Testing
Software Quality Testing	Software Quality Testing (See Note 2)

Note 1 - Subsystem Design Certification Testing was conducted by T&E Engineering. The testing was conducted to verify the requirements of the Ship Based Electronics Subsystem, which includes the seven operational Computer Program Configuration Item's. It has been included in this listing for completeness only.

Note 2 - Software Quality Testing, a requirement of MIL-STD-1679 for program acceptance, was conducted by T&E Engineering as a part of the system reliability demonstration test. It has been included in this listing for completeness only.

3.0 Correlated Requirements of MIL-STD-1679 and Implementation in SDP

The following sections relate the explicit requirements of MIL-STD-1679 to the responses to them in the SDP. Each section reiterates the MIL-STD-1679 requirement and is followed by the SDP response. For reference purposes the MIL-STD-1679 and SDP paragraph numbers have been included as parenthetical comments.

3.1 MIL-STD-1679 Introduction

This section has been included for completeness. It states the overall test philosophy of the Standard. Since it is an introductory section, no explicit SDP sections relate to it.

3.1.1 Program Test (MIL-STD-1679 paragraph 5.8)

The contractor shall determine the scope of tests required to ensure that the program being developed meets all specified technical, operational, and performance requirements and the acceptance criteria. The contractor shall be responsible for accomplishing all development testing. Test planning shall include development of:

a. Program acceptance criteria.

b. Levels of testing to verify performance.

c. Internal procedures for scheduling and conducting tests.

d. Detailed procedures for testing at each level.

e. Reporting procedures of test results.

All tests plans, specifications, and procedures shall be subject to review and approval by the procuring agency.

The contractor shall provide or ensure the availability of adequate facilities for conducting all required tests. The procuring agency shall have the option of specifying the facility that should be used to conduct any portion of the test program.

The contractor shall prepare test reports showing quantitative results of all tests. Testing shall consist of the following:

a. Module tests.

b. Subprogram tests.

c. Program performance tests.

d. System(s) integration tests.

3.2 Module Tests and Subprogram Tests

3.2.1 MIL-STD-1679 Requirement

3.2.1.1 Module Tests (MIL-STD-1679 paragraph 5.8.1)

Each module shall have completed a code walk-through prior to being subjected to developmental testing. Developmental testing shall be adequate to determine compliance with the applicable technical, operational and performance specifications. As a minimum, module testing shall be performed to:

a. Ensure error-free compile/assembly of the coded module.

b. Ensure that the coded module fully satisfies the detailed performance and design requirements and that all code to be delivered has been exercised.

c. Exercise the module in terms of input/output performance with the results satisfying the applicable detailed performance and design requirements.

3.2.1.2 Subprogram tests (MIL-STD-1679 paragraph 5.8.2)

Modules shall have passed the module tests prior to being subjected to subprogram testing. The modules shall be integrated individually into particular subprograms. Subprogram tests shall be adequate to determine compliance with the applicable technical, operational and performance requirements. As a minimum, subprogram testing shall be performed to:

a. Ensure error-free linkage of the modules.

b. Ensure that the subprogram fully satisfies the detailed performance and design requirements.

c. Exercise the subprogram in terms of input/output performance with the results satisfying the applicable detailed performance and design requirements.

d. Ensure the subprogram level man-machine interfaces.

e. Ensure the capability of the subprogram to handle properly and survive erroneous inputs.

3.2.2 SDP Response

3.2.2.1 Levels of Testing (SDP Paragraph 4.1.1)

Four levels of software testing will be conducted prior to formal level 5 Subsystem Design Certification testing being performed by T&E Engineering. The first test level, unit and subprogram testing, will be conducted by the software development team, followed by build, validation, and integration testing conducted by the software test team.

3.2.2.2 Unit Testing (SDP Paragraph 4.2)

The concept of top-down unit and subprogram testing requires that the top level control units of each subprogram be coded first with suitable stubs provided for lower level units. Partially stubbed subprograms may then be plugged into the dummy process so that higher level units become drivers for lower level units and the software is tested with actual interfaces being exercised. For each subprogram, Level 1 test cases and test results are documented in the Program Development Notebooks PDN's).

Unit testing also includes isolated tests with special drivers as necessary, testing over a wide range of input parameters (both legal and illegal), test of all error responses, test responses, test of all program logic switches, and all independent logic paths. Much of the software coming from the design methods, described in paragraph 3.2, will lend itself to unit testing by implementing simple functions together with a few items of data set in the data base; and when this is the case, this method of testing will be used.

The software test bed, which consists of the initial test bed software plus integrated builds, will be used as a unit test driver for those units requiring a dynamic environment. Structured and top down implementation, and careful build designs will minimize the amount of special test software required for unit testing.

Testing of a unit of software tests the requirements allocated to that unit. Those cases where a requirement is unique to a given unit and is fully verified by unit testing will be clearly marked as such on the traceability matrix. In those cases where a particular requirement is allocated to more than one unit the requirement must be verified at the subprogram level or the build level testing.

Unit testing responsibilities of the software development team include integrating each unit into the evolving test bed to provide confidence that when a group of units are delivered, they will operate in the integrated build environment. This approach will also leave the software under the control of the developers until development is complete and it is ready for turnover. Once delivered and turnover tests are successfully run the build contents are placed under configuration control, and each build serves as a baseline for all subsequent development.

A build turnover test with clearly specified pass/fail criteria will be designed by the test team with participation by the development group responsible for the software. The idea of the turnover test is to establish sanity of the build in the test bed. This means that the software will cycle and that a few high level capabilities will be satisfied in order to facilitate further testing. The pass criteria will not be rigorous for turnover tests. The strategy is for the developer to pass the turnover test in the development test bed, then deliver the software to the test team. Once the test team certifies that the software has passed the turnover test, it becomes their responsibility and the development teams from that point on need only respond to software problem reports against this software.

3.3 Program Performance Tests

3.3.1 MIL-STD-1679 Requirement

3.3.1.1 Program Performance Tests (MIL-STD-1679 Paragraph 5.8.3)

Program performance tests. All subprograms shall have passed the subprogram tests prior to program performance testing. The subprograms shall be integrated individually until all subprograms have been integrated into the program. These tests shall be adequate to determine compliance with the applicable technical, operational and performance requirements. As a minimum, program performance testing shall be performed to:

a. Ensure the total man-machine interface.

b. Ensure proper system initiation, data entries via peripheral devices, program loading, restarting, and the monitoring an controlling of system operation from display consoles and other control stations as applicable.

c. Ensure the proper interfacing of all equipment specified in the program performance requirements.

d. Ensure the capability of the program to satisfy all applicable system and program performance requirements.

e. Ensure the capability of the system to handle properly and survive erroneous inputs.

3.3.2 SDP Response

3.3.2.1 Build Testing - Level 2 (SDP Paragraph 4.3)

At the completion of unit and subprogram testing for a particular build, the Software Design Team will prepare a Software Delivery Report for the Software Test Team containing detailed information regarding the content of the build. The Software Test Team leader does not accept the delivery until a series of regression tests to check previous capabilities and turnover tests have been successfully completed by the Test Team. At this time, the code is placed under configuration control, and problem reporting procedures are implemented as specified in the Software Standards and Procedures Manual (SS&PM).

After successfully conducting the turnover tests, the test team will proceed with the level 2 build tests in accordance with test specifications and procedures previously approved by the software test review board. The software test review board will be chaired by the AN/SQR-19 Software Engineering Manager or his delegate and have representatives from Systems Engineering, Software Engineering, T&E Engineering, Quality Assurance, and the Navy.

Build testing is aimed at testing integrated strings of software units concentrating on subprogram and Computer Program Configuration Item (CPCI) interfaces, and the inputs and outputs to the strings, as opposed to individual parameter tests for code units. Each build will be exercised to verify the added functions, and will be regression tested to show that the added software did not degrade previous capabilities.

For each software build, the software test team will compile a Build Test Notebook (BTN), containing support information and historical data generated during the test period. The BTN shall contain weekly status review minutes, software delivery reports, tape and disk assignment numbers, red-lined test procedures, test execution reports, and test results. At the conclusion of build testing, a copy of the test report will be placed in the BTN, and the BTN will become a historical record of the build test activity.

The approach that will be used to drive the software during build and validation testing will be to periodically load data into the input buffers from a test driver computer which will read the data from a magnetic tape or disk.

Scenario tapes will be generated to reflect the input header and data formats. Target data will be nominal with no attempt to simulate position error on targets accurately but data rates will, in most cases, be accurately simulated. The software to generate these data tapes and interface with the real-time software will be developed by the Process Design Integration and Support Software Group. The tape generation program will be run off-line. The input interface program will be run in real time.

3.3.2.2 Validation Testing - Level 3 (SDP Paragraph 4.4)

The Computer Program Test Plan specifies that Level 3 Validation Testing will be conducted in the Ship-based Electronics Subsystem (SES) Test Facility (STF) and Integrated System Test Facility (ITF) facilities in accordance with deliverable test specifications and test procedures. All test documentation will be subject to approval by the software test review board. The test configuration for validation testing will be similar to that used for the final build for each CPCI.

3.4 System(s) Integration Test

3.4.1 MIL-STD-1679 Requirements

3.4.1.1 System(s) Integration Test (MIL-STD-1679 Paragraph 5.8.4)

In instances where the developed program is an element of a larger system involving the integration of two or more programs, the contractor(s) shall be required to participate in total system integration testing. Integration testing may be conducted at facilities other than the development facility, such as a land-based test site. Each contractor shall provide technical support to the integration testing as required.

3.4.2 SDP Response

3.4.2.1 Software Integration (Level 4) (SDP Paragraph 10.0)

Software integration will be accomplished to verify interface performance of the AN/SQR-19 computer programs in the SES real time multi-computer environment. During software integration five integration threads will be generated to verify the interrelationships between builds of different CPCI's just as incremental builds will be used to verify the interrelationships between subprograms of individual CPCI's. By decomposing the software into these five incremental threads, problem areas can be identified earlier in the development cycle and risks associated with inter-computer communications are reduced. Upon completion of software integration, all of the incremental builds of the An/SQR-19 CPCI's will have been integrated and tested into the AN/SQR-19 real time multi-computer software subsystem.

3.4.2.2 Integration Approach (SDP Paragraph 10.1)

Integration of CPCI builds into threads will be performed by the Process Design Integration and Support Software Group. This group will construct the threads and assure that they can be loaded and will cycle in the SES computers before delivering the thread to the Program Test Group for integration testing. Early threads will consists of predefined CPCI builds and dummy data generators, providing the ability to pass data between and through the programs. Additional capability will be included in subsequent threads to process data and to transfer data between programs leading to full functional capability for all CPCI's in Thread 5 (final integration software configuration).

3.4.2.3 Program Integration (SDP Paragraph 10.1.1)

During program integration the Process Design and Integration Team will combine predetermined builds of each CPCI into load modules called threads for the AN/SQR-19 multi-computer system. As each build is turned over, it will be integrated with other CPCI builds until all builds specified have been integrated into the software thread.

For each thread, the Process Design and Integration Team will load the computer system, and exercise the operational software in real time to check timing relationships between programs and inter-computer messages. Once this "sanity check" is demonstrated and the thread is able to cycle, the Integration Team will turnover the thread to the independent test team for integration testing.

3.4.2.4 Thread Testing (SDP Paragraph 10.1.2)

Verification of interface performance requirements contained in the Program Performance Specifications, Interface Design Document and Interface Design Specification (partial) will take place during integration thread tests. A systematic method of requirement traceability will be performed for thread testing using a traceability matrix technique. Each requirement entered on the matrix will be reviewed by software designers and the test team to determine which requirements are to be verified at the integration test level (Level 4). Particular attention will be placed on assuring that the matrix is updated to reflect requirements changes and that appropriate retest of a subsequent design change is performed. Traceability of requirements to specific test cases will be included in each thread test specification. Test cases will be designed to test inter-CPCI data and control transfers and will include tests of data paths through the operational CPCI's.

Test specifications and test procedures will be prepared and reviewed prior to the start of each thread test. Test reports will be prepared at the conclusion of each thread test.

Appendix B

TACTAS Build Test Approach

1.0 Overview

This section reiterates the build test approach employed on TACTAS as described in the Software Development Plan.

2.0 Test Approach

Top down testing using a series of incremental builds is the key element in the approach to development and validation of each of the AN/SQR-19 CPCI's. This approach has been proven to be an efficient method of integrating software units and subprograms into a working CPCI and verifying that performance and design requirements have been met.

The preliminary software build, which is used as a test bed by the Software Design Team for lower level unit and subprogram testing, is called the dummy process. The dummy process initially contains dummy stubs for all application subprograms which are subsequently replaced by the actual code through a series of incremental software builds. Build 1 testing, for example, is conducted for each CPCI by the software test team when lower level tests are completed by the design team on the units and subprograms comprising the build 1 capability. As described in subsequent paragraphs, the build 1 capability will be augmented in subsequent builds by additional code deliveries for each CPCI. At the conclusion of the final build test (which is conducted with all subprograms in a given CPCI), validation testing will be performed on the CPCI.

3.0 Levels of Testing

Four levels of software testing shown in Table B-1 will be conducted prior to formal level 5 Subsystem Design Certification testing being performed by T&E Engineering. The first test level, unit and subprogram testing, will be conducted by the software development team, followed by build, validation, and integration testing conducted by the software test team. Acceptance of code from the development team by the software test team will be on the basis of pre-defined build turnover tests. T&E Engineering will accept the software from the Software Test Team after successful completion of validation and integration testing.

As shown in Table B-1 all levels of testing will be fully documented by the group responsible for conducting the tests and frequent QA audits will be conducted at all test levels. A software test review board will verify that all planned tests have been conducted and that the results of these tests reflect successful conduction of tests specified in the validation test specifications. A description of each level of testing to be conducted appears in subsequent paragraphs of this section.

4.0 Requirements Verification/Validation

Figure B-1 illustrates how PPS requirements allocated by the PDS to units, subprograms, or builds are verified at the test levels shown.

Level 1 unit and subprogram testing is driven by Functional Capability Lists (FCL's) generated by the software designers and included in the PDN's. These FCL's include PPS requirements and PDS design requirements which can be verified during Level 1 testing.

For each build, the software test team generates FCL's which are the driving requirements for the build test specifications/test procedures. These FCL's include PPS requirements which can be verified during Level 2 testing.

The driving requirements for Level 3 validation testing, which is conducted on the fully integrated CPCI, are specified in Section 4.0 of the PPS.

Integration testing Level 4 driving requirements are the detailed intra-CPCI interface specifications contained in the IDD, the IDS and the PPS.

The requirements to be verified during Subsystem Design Certification testing (Level 5) which is conducted by T&E Engineering, are contained in Section 4.0 of the SES B1 performance specification and Section 4.0 of the SES B1 development specification.

A Systematic method of requirement traceability will be performed for each CPCI using a PPS requirements traceability matrix. Each requirement entered on the matrix is reviewed by the software designers and the test team to determine if the requirements are valid, and whether they are verifiable at the unit, subprogram, build, validation or integration test level. Particular attention will be placed on assuring that the matrices are updated to reflect requirements changes and that appropriate retest of a subsequent design change is performed. Traceability of requirements to specific test case will be included in each build, validation and integration test specification and will also be included in the PDN.

TABLE B-1

TEST LEVEL DEFINITION AND DOCUMENTATION

Test Level	Documentation	
Level 1 - Unit and Subprogram Testing - Requirements Allocated to Units are Verified - Design requirements in PDS are verified	Program Development Notebook - Requirements Traceability Matrix - Test Cases - Test Results	SW Development Team QA Audit
Level 2 - Build Testing - Requirements Allocated to Groups of Units are Verified	Test Specs Procedures Report	SW Test Team QA Audit
Level 3 - Validation Testing - Verification of PPS Requirements in Accordance with Section 4.0	Test Specs Procedures Report	SW Test Team/T&E Witness Navy & QA Witness
Level 4 - Software Integration Testing - Verification of PPS and IDD interface requirements	Test Specs Procedures Report	SW Test Team/T&E Witness Navy & QA Witness
Level 5 - Subsystem Design Certification Testing - Verification of SES B1 Performance Spec Requirements - Verification of SES B1 Development Spec Requirements	Test Specs Procedures Report	T&E Engineering Navy & QA Witness

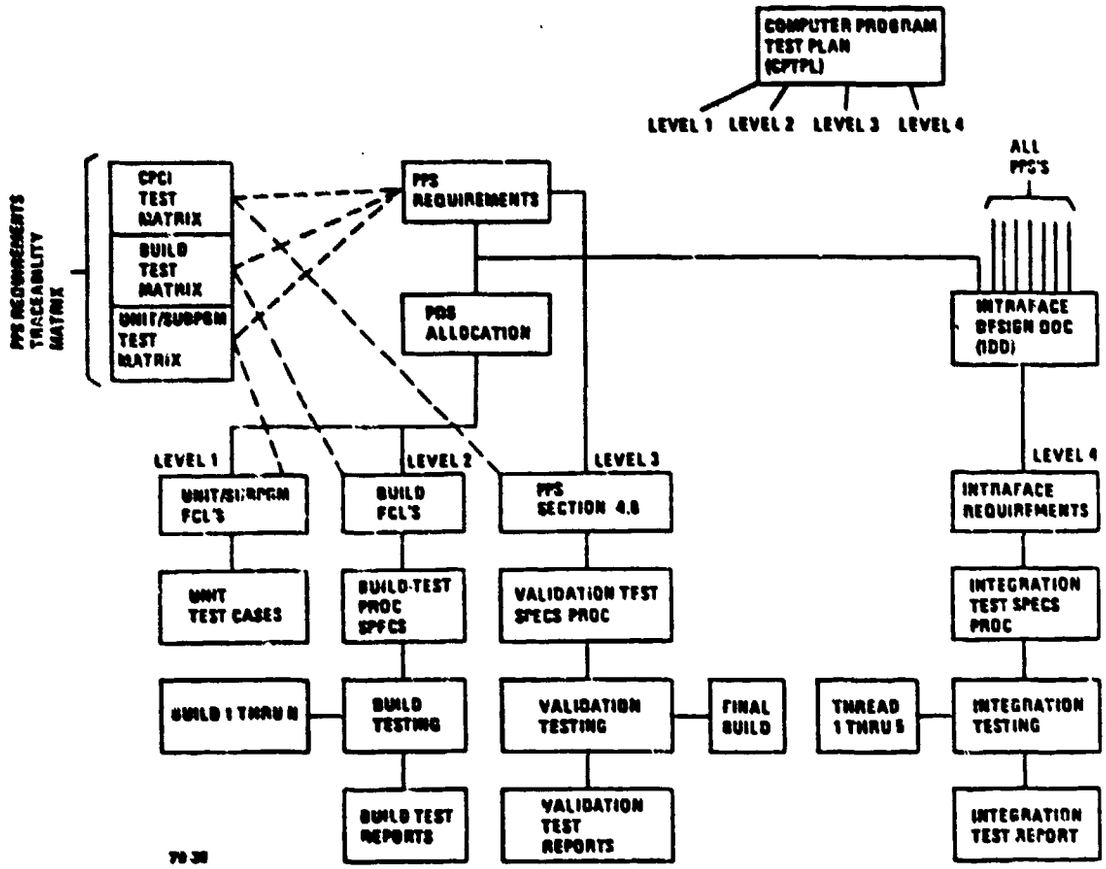


Figure B-1 Software Requirements/Verification Traceability

5.8 Unit Testing

The concept of top-down unit and subprogram testing requires that the top level control units of each subprogram be coded first with suitable stubs provided for lower level units. Partially stubbed subprograms may then be plugged into the dummy process so that higher level units become drivers for lower level units and the software is tested with actual interfaces being exercised. For each subprogram, Level 1 test cases and test results are documented in the Program Development Notebooks (PDN's).

Unit testing also includes isolated tests with special drivers as necessary, testing over a wide range of input parameters (both legal and illegal), test of all error responses, test responses, test of all program logic switches, and all independent logic paths.

Unit testing responsibilities of the software development team include integrating each unit into the evolving test bed to provide confidence that when a group of units are delivered, they will operate in the integrated build environment. This approach will also leave the software under the control of the developers until development is complete and it is ready for turnover. Once delivered and turnover tests are successfully run the build contents are placed under configuration control, and each build serves as a baseline for all subsequent development.

A build turnover test with clearly specified pass/fail criteria will be designed by the test team with participation by the development group responsible for the software. The idea of the turnover test is to establish sanity of the build in the test bed. This means that the software will cycle and that a few high level capabilities will be satisfied in order to facilitate further testing. The pass criteria will not be rigorous for turnover tests. The strategy is for the developer to pass the turnover test in the development test bed, then deliver the software to the test team. Once the test team certifies that the software has passed the turnover test, it becomes their responsibility and the development teams from that point on need only respond to software problem reports against this software.

6.0 Build Testing (Level 2)

At the completion of unit and subprogram testing for a particular build, the Software Design Team will prepare a Software Delivery Report for the Software Test Team containing detailed information regarding the content of the build. The Software Test Team leader does not accept the delivery until a series of regression tests to check previous capabilities and turnover tests have been successfully completed by the Test Team. At this time, the code is placed under configuration control, and problem reporting procedures are implemented as specified in the Software Standards and Procedures Manual (SS&PM).

After successfully conducting the turnover tests, the test team will proceed with the level 2 build tests defined in Table B-1, in accordance with test specifications and procedures previously approved by the software test review board.

Build testing is aimed at testing integrated strings of software units concentrating on subprogram and CPCI interfaces, and the inputs and outputs to the strings, as opposed to individual parameter tests for code units. Each build will be exercised to verify the added functions, and will be regression tested to show that the added software did not degrade previous capabilities.

For each software build, the software test team will compile a Build Test Notebook (BTN), containing support information and historical data generated during the test period. The BTN shall contain weekly status review minutes, software delivery reports, tape and disk assignment numbers, red-lined test procedures, test execution reports, and test results. At the conclusion of build testing, a copy of the test report will be placed in the BTN, and the BTN will become a historical record of the build test activity.

The approach that will be used to drive the software during build and validation testing will be to periodically load data into the input buffers from a test driver computer which will read the data from a magnetic tape of disk.

Scenario tapes will be generated to reflect the input header and data formats. Target data will be nominal with no attempt to simulate position error on targets accurately but data rates will, in most cases, be accurately simulated. The software to generate these data tapes and interface with the real-time software will be developed by the Process Design Integration and Support Software Group. The tape generation program will be run off-line. The input interface program will be run in real time.

7.0 Validation Testing (Level 3)

The Computer Program Test Plan specifies that Level 3 Validation Testing will be conducted in the STF and ITF facilities in accordance with deliverable test specifications and test procedures. All test documentation will be subject to approval by the software test review board. The test configuration for validation testing will be similar to that used for the final build for each CPCI.

8.0 Software Integration Testing

The Computer Program Test Plan (CPTPL) specifies that software integration testing of the six AN/SQR-19 CPCI's will be conducted to verify intercomputer message requirements of the IDD, PPS's, and PDS's. Integration testing will be conducted on a series of five integration threads of the AN/SQR-19 software. Each thread will include increasing intercomputer functions leading to complete intercomputer system software functions in thread 5. At the completion of integration thread testing and validation testing the AN/SQR-19 will be turned over to T&E Engineering for Subsystem Design Certification testing.

9.0 Subsystem Design Certification Testing

Subsystem Design Certification Testing will be conducted in accordance with the Master Test & Evaluation Plan to verify requirements specified in Section 4.0 of the SES B1 Development Specification.

Software Engineering will be in a support role during System Design Certification Tests to be conducted by T&E Engineering.

INDEX